Device Detection APIs

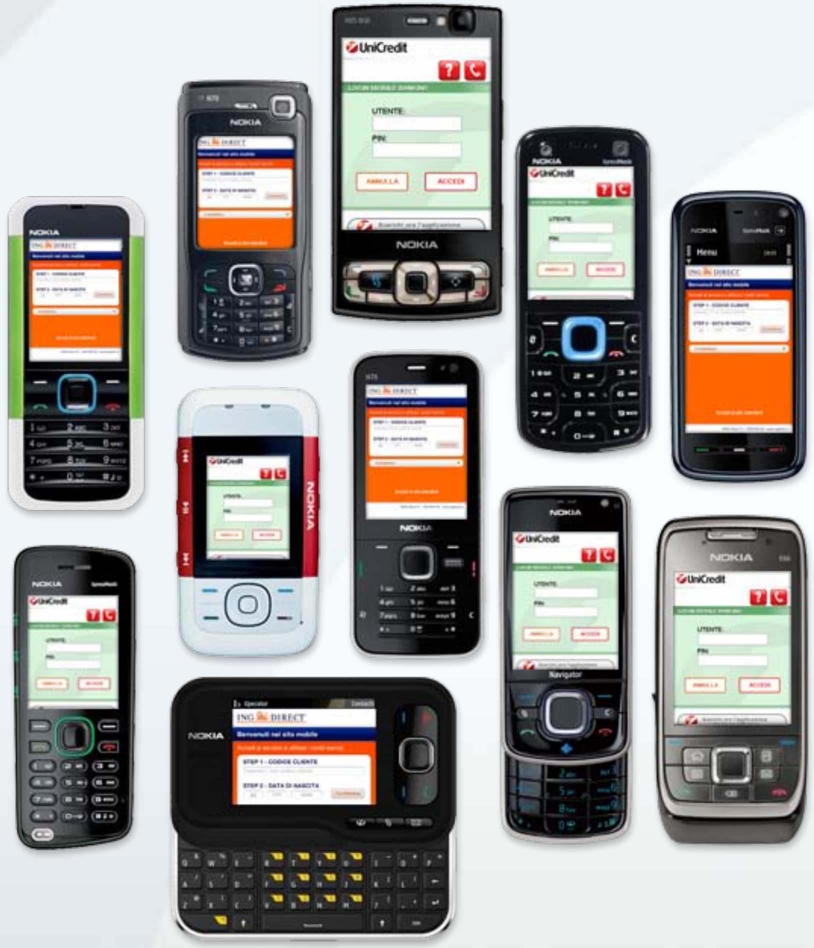
Proposal for a new JSR on top of JEE

Stefano Andreani

Java Community Process

Java Community Process

Background on device detection (1/2)



Identifying the capabilities of a device accessing web contents has been an extensively explored issue in the past years, in particular in the mobile browsing space, where multiple different standards have been coexisting in the market *(WML, Compact HTML, XHTML Mobile Profile, and various sets of scripting languages)*.

The focus at that stage was identifying supported standards and screen size, in order to be able to adapt the contents.

Background on device detection (2/2)

The recent diffusion of new connected devices, like tablets and connected TVs, makes the task of device detection more complex, because identifying supported standards is not enough to create an effective application: the possible interaction modes are extremely important for the developer to create an optimized user experience.

Solutions from the industry and the community

Several solutions exists from the community and from the industry. The WURFL project (Wireless Universal Resource File) is the most famous and widely adopted solution coming from the community. It provides a repository of devices capabilities and a GPL licensed API to access the repository.

With similar purposes Oracle designed the "ADF Mobile Browser" and IBM the "Mobile Portal Accelerator”, both adding content adaptation capabilities to device detection (just to mention two).

About the ADF Mobile Browser

Oracle Application Development Framework Mobile (ADF Mobile) browser is a standards-based framework that enables the rapid development of enterprise mobile applications. Oracle Fusion Middleware 11g release 1 of ADF Mobile browser extends Oracle ADF to browsers running on mobile devices. Because ADF Mobile browser is built upon the component model of Java Server Faces (JSF), you can quickly develop applications for mobile browsers. ADF Mobile browser's mobile-specific extensions to JSF enable you to develop mobile applications using the same methodologies for developing JSF applications for the desktop.

When developing an ADF Mobile browser application, you need not focus on the limitations or capabilities of different browsers, as ADF Mobile browser enables you to develop applications that function properly on different browser types. The ADF Mobile browser renderer ensures that contents can be consumed correctly by the target browser.

About the ADF Mobile Browser

How to Determine Browser Capabilities
ADF sends its response to a user-agent's request based on capabilities it assigns to a user agent. These capabilities include a user-agent's support for JavaScript, PPR, and so on. Some of these capabilities are exposed to developers through the EL expression #{requestContext.agent.capabilities}.

Browser Capabilities Exposed through EL Expressions:

narrowScreen
: Indicates whether ADF rendering engine optimizes his rendering for a narrow-screen device.

scriptingSpeed
: Indicates JavaScript support for a user-agent. Returns "none"(a String type) if the user-agent does not support JavaScript.

partialRendering
: Indicates PPR support for a user-agent. Returns true (a boolean type) if the browser supports PPR.

IBM Mobile Portal Accelerator

IBM Mobile Portal Accelerator software is an advanced content adaptation product that helps you customize portal content for mobile devices, providing mobile users access to portals through a highly navigable, personalized Web experience. Business services, content, and commerce applications can be accessed from virtually anywhere using the same open, Web-based integrated business services traditionally used on laptop and desktop computers provided through WebSphere Portal software.

This product helps extend the capabilities of WebSphere Portal to mobile devices by enabling "write once, render many" through device-independent Web authoring for limited capability devices, such as XHTML, HTML, WML, and cHTML for imode-based mobile devices. It can provide an easy, cost-effective way to deliver content and applications to many mobile devices. For example, developers can create one version of Web content and use it for multiple mobile devices rather than developing content for each device. The Mobile Portal Accelerator software can adapt the content for each mobile device based on the specific device characteristics and capabilities, which are defined in the software's device repository.

The IBM Mobile Portal Accelerator Device Update service offering is provided as a subscription service that enables updated device profile information to be added to the device repository files that are delivered with IBM Mobile Portal Accelerator. New devices can be introduced through the update service without changes to style guides and without impacting existing applications or products.

IBM Mobile Portal Accelerator

Device Capabilities

Mobile Portal provides a set of rules based on device capability attributes that govern when to display a node within the navigation tree.

Mobile Portal supports the following device capabilities: Image capable

- •Color capable
- •Sound output capable
- •Client accept types –accepted formats for images, video, sound, and other formats.

Mobile Portal also provides the infrastructure for defining customized device capabilities. The following device capabilities are examples of custom attributes that may be added to the attribute descriptor file. Full support for some of these capabilities may require custom aggregator changes. This list is not exhaustive and other custom attributes can be added.

- •Screen size
- •Browser name
- •Browser version
- •Screen size char
- •WML version
- •Bits per pixel
- •Horizontal scroll
- •Vertical scroll
- •Portrait or landscape orientation
- •Camera
- •File download

WURFL APIs

WURFL APIs are the community supported way to access WURFL information. The WURFL is an XML configuration file which contains information about capabilities and features of many mobile devices.

The project page contains the following disclaimer:
There is no guarantee of any kind that any of the info in the WURFL is correct. All the information has been gathered by collecting reports from users and developers around the globe.

WURLF APIs define the **Device** type which will allow to retrieve all the information about a device and its capabilities:

java.util.Map **getCapabilities**()	Returns a device's capabilities name-value map.
java.lang.String **getCapability**(java.lang.String name)	Return a device's capability value.
java.lang.String **getId**()	Return the WURFL id of this device
java.lang.String **getUserAgent**()	Return the user-agent string for this device as defined in WURFL.

Limitations of existing solutions

JEE applications using one of the mentioned solutions are designed with a strong dependency on the underlying device database and on the APIs designed to access it.

Each solution defines a different way to manage the update of the list of devices, and capabilities are changed following market evolution.

These isseus cause fragmentation in source code and in development and deployment process. There's no way to implement a portable multi-device Web application.

JSR Section 2: Request (1/6)

Q1. Please describe the proposed Specification.

The Device Detection specification will extend the Servlet API by providing a standard way to retrieve device capabilities: JEE applications will identify the capabilities of the accessing client, and will manage information flow, business logic and user-experience according to client capabilities.

In order to achieve this, the proposed specification will define a new `Device` interface and a new `ServletRequest method:` Device getDevice( )

JSR Section 2: Request (2/6)

Aim of the Expert Group will be to define and formalize the Device interface. From a preliminary analysis the Device interface should manage the following set of device capabilities:

- Size in inch: size of the main browser window in full screen- density in dpi: density of the screen
- Tagging support (list of supported tagging standards and versions)
- Scripting support (list of supported scripting standards and versions)
- Pointing methods: rocker, touchscreen, wheel, stylus.

Additional device capabilities may be retrieved depending on implementation

JSR Section 2: Request (3/6)

This first version of the Device Detection specification will concentrate on the following design goals:

- Independency of capability discovery from the sources from which capabilites are retrieved.
- Simple device classification.

The Objective of this JSR is to create a standard for Client-aware Applications, which will help unifying a fragmented area and will allow the development of server-side-portable multidevice applications.
The expert group will ensure this specification draws appropriately from existing projects and products and that it will be based on open standards. Content adaptation is not covered by [the first version of] this JSR, in order to keep it small and for guaranteeing short time to market. [It could be planned in a following stage.]

JSR Section 2: Request (4/6)

Q2. What is the target Java platform?

A Java extension for the JEE 5 platform.

Q3. What need of the Java community will be addressed by the proposed specification?

This specification will establish a standard API for retrieving capabilities of clients accessing contents via http requests, thus avoiding locking applications on a specific JEE implementation and allowing software developers to reach a wider audience while reducing their development efforts.

The Device Detection specification is required to achieve independency of capability discovery from the sources from which capabilites are retrieved. The goal is to allow Java web applications to retrieve device capabilities in a standard way on any server implementing the specification.

Q4. Why isn't this need met by existing specifications?

The Servlet/JSP specifications don't define any stardard way to retrieve device capabilities.

JSR Section 2: Request (5/6)

Q5. Please give a short description of the underlying technology or technologies:

The Device Detection specification will be designed leveraging the following technologies: XML, Servlet/JSP and other JEE technologies.

For example, a JSP tag library extension could be used by a software developer to render the requested resource.

Q6. Is there a proposed package name for the API Specification?

javax.servlet.deviceDetection

Q7. Does the proposed specification have any dependencies on specific operating systems, CPUs, or I/O devices that you know of?

No.

Q8. Are there any security issues that cannot be addressed by the current security model?

No

JSR Section 2: Request (6/6)

Q9. Are there any internationalization or localization issues?

No

Q10. Are there any existing specifications that might be rendered obsolete, deprecated, or in need of revision as a result of this work?

No.

Q11. Please describe the anticipated schedule for the development of this specification.

To be determined by the expert group, the target is to keep the API at a minimum to keep a tight schedule.

Q12. Please describe the anticipated working model for the Expert Group working on developing this specification.

The spec leader should come from the development community, with experience on mobile web application development. This would allow to keep the developers' perspective and to consider all the main existing solutions as a basis for this JSR. A must to start the activity is having within the EG major JEE implementors.
We anticipate a mixture of mailing list and occasional face to face or teleconference meetings.